Analysis of Hash Functions and Cellular Automata Based Schemes

Jun-Cheol Jeon

Department of Computer Engineering,
Kumoh National Institute of Technology, Gumi, Korea
jcjeon@kumoh.ac.kr

Abstract

In this paper, we summarize hash functions and cellular automata based architectures, and discuss some pros and cons. We introduce the background knowledge of hash functions. The properties and theory of cellular automata are also presented with typical works. We show that cellular automata based schemes are very useful to design hash functions with a low hardware complexity because of its logical operation attributes and parallel properties.

Keywords: *cryptography, hash function, cellular automata, hardware complexity*

1. Introduction

Any message authentication or digital signature mechanism can be viewed as having fundamentally two levels [1]. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

In order to authenticate a message, the types of functions may be grouped into three classes: message encryption (the ciphertext of the entire message serves as its authenticator), message authentication code (MAC, a public function of the message and a secret key that produces a fixed-length value that serves as the authenticator), and cryptographic hash function (a public function that maps a message of any length into a fixed-length hash value, which serves as the authenticator).

This paper is only concerned with cryptographic hash functions which are classified into two classes: unkeyed hash function (hash functions, whose specification dictates a single input parameter - a message) and keyed hash function (hash functions, whose specification dictates two distinct inputs - a message and a secret key). This paper is only concerned with unkeyed hash function which is also called one-way hash function.

Cryptographic hash functions play an important role in modern cryptography. The basic idea of cryptographic hash functions is that a hash-value serves as a compact representative image (sometimes called an imprint, digital fingerprint, or massage digest) of an input string, and can be used as if it was uniquely identifiable with that string [2].

From a structural point of view, cryptographic hash functions may be categorized based on the nature of the operations comprising their internal compression functions. From this viewpoint, there are several broadcast categories of iterated hash functions studied to date are the functions based on block ciphers, the functions based on modular arithmetic, and dedicated functions. Specially, dedicated functions are those designed specially for hashing, with speed in mind and independent of other system subcomponents (*e.g.*, block cipher or modular multiplication subcomponents which may already be present for non-hashing purposes)

The following cryptographic hash functions, all based on the so called MD4 initially proposed in [3] have received the greatest attention: MD5 [4], SHA-1 and SHA-256 [5], RIPEMD-160 [6], and HAVAL [7]. However, in applications where speed is important and very large amounts of data have to be authenticated (*e.g.*, electronic financial transactions, software integrity), hardware implementations are the natural solution. Thus dedicated cryptographic hash functions based on cellular automata are strongly recommended.

Cellular Automata (CA) are among the oldest model of natural computing, dating back over half a century. The first CA studies by John von Neumann in the late 1950s were biologically motivated [8]: the goal was to design self-replicating artificial systems that are also computationally universal. A CA possesses several fundamental properties of the physical world: they are massively parallel, homogeneous and all interactions are local. Other physical properties such as reversibility and conversion laws can be programmed by choosing the local update rule properly. It is therefore not surprising that physical and biological systems have been successfully simulated using CA models [9]. The physical nature of CA may have even much greater practical importance when applied to the opposite direction, that is, when using the physics to simulate CA. Since many CA are computationally universal and some very simple CA have this property then perhaps we eventually succeed to harness physical reactions of microscopic scale to execute massively parallel computations by running a computationally universal CA. This requires that the simulated CA obeys the rules of physics, including reversibility and conservation laws. While such truly programmable matter may be decades away, its potential is great [10].

Wolfram [11] pioneered the investigation of CA as mathematical models for self-organizing statistical systems and suggested the use of a simple two-state, three-neighborhood CA with cells arranged linearly in one dimension. CAs are dynamic systems in which space and time are discrete. A CA consists of an array of cells, each of which can be in one of a finite number of possible states, updated synchronously in discrete time steps, according to a local, identical interaction rule. Martin, *et al.*, [12] used polynomial algebraic tools to derive some characterizations of uniform CA with identical rules applied to each of the cells. Subsequently, Das, *et al.*, [13, 14] developed a matrix algebraic tool capable of characterizing hybrid CA with different rules applied to different cells.

In the last two decades, a wide variety of applications on CA has been proposed. Major applications can be categorized under the following broad headings: simulation of physical systems, biological modeling involving models for self-reproduction, image processing, language recognition, computer architectures, error correcting codes, block and stream cipher cryptography, and fractals and chaos.

In [15], Daemen, *et al.*, have persisted in vulnerability of scheme from [16] together with a new CA based hash function. Another research on CA based hash function has been reported by Mihaljevic, *et al.*, [17] based on their previous report in [2]. They have proposed a family of fast dedicated one-way hash functions based on linear CA over GF(q). Recently, Jeon proposed one-way hash function using linear and nonlinear CA.

One thing we know is that the security of hash functions is indeed based on confusion and diffusion. However, it is quite hard to explain a level of confusion and diffusion so that experimental results should be provided and compared with the previous well-known hash functions.

The purpose of this paper is to find characteristics of CA and design a secure and efficient cryptographic hash function using simple CA operations. An important feature of the proposed one-way hash function is that it is especially suitable for compact and fast implementation in hardware. The proposed scheme provides outstanding efficiency in cryptographic application use. The proposed scheme is a CA based hash function for an arbitrary length message hashing, called CAH-256 that the scheme takes as input a message with a maximum length of less than 2^{64} bits and produces as output a 256-bit message digest.

2. Preliminary

In this section, we introduce the background knowledge useful in understanding the hash function and cellular automata. In Section 2.1, we present some definitions and security requirements of cryptographic hash function. Section 2.2 presents some basic concepts of cellular automata. Moreover, the previous cryptographic hash schemes are briefly reviewed in Section 2.3.

2.1. Cryptographic Hash Functions

A cryptographic hash function can provide assurance of data integrity. A hash function is used to construct a short "fingerprint" of some data; if the data is altered, then the fingerprint will no longer be valid. Even if the data is stored in an insecure place, its integrity can be checked from time to time by recomputing the fingerprint and verifying that the fingerprint has not changed [18].

2.1.1. Security of Hash Functions: Suppose that H: X $\rightarrow$ Y is an unkeyed hash function. Let $x \in$ X, and define $y = H(x)$. In many cryptographic applications of hash functions, it is desirable that the only way to produce a valid pair (x, y) is to first choose x, and then compute $y = H(x)$ by applying the function H to x. Other security requirements of hash functions are motivated by their applications in particular protocols, such as signature schemes. We now define three problems; if a hash function is to be considered secure, it should be the case that these three problems are difficult to solve [18].

Problem 1: *Preimage(one-way)*
Instance: A hash function H: X $\rightarrow$ Y and an element $y \in$ Y.
Find: $x \in$ X such that $H(x) = y$.

Given a possible message digest y, the problem preimage asks if x can be found such that $H(x) = y$. If preimage can be solved for a given $y \in$ Y, then the pair (x, y) is a valid pair. A hash function for which preimage cannot be efficiently solved is often said to be one-way or preimage resistant.

Problem 2: *Second Preimage(weakly collision)*
Instance: A hash function H: X $\rightarrow$ Y and an element $x \in$ X.
Find: $x' \in$ X such that $x' \neq x$ and $H(x') = H(x)$.

Given a possible message x, the problem second preimage asks if $x' \neq x$ can be found such that $H(x') = H(x)$. Note that, if this can be done, then $(x', H(x))$ is a valid pair. A hash function for which second preimage cannot be efficiently solved is often said to be second preimage resistant.

Problem 3: *Collision*
Instance: A hash function H: X $\rightarrow$ Y.
Find: $x, x' \in$ X such that $x' \neq x$ and $H(x') = H(x)$.

The problem collision asks if $x' \neq x$ can be found such that $H(x') = H(x)$. A solution to this problem does directly yield a valid pair. However, if (x, y) is a valid pair and x, x' is solution to collision, then (x', y) is also a valid pair. There are various scenarios where we want to avoid such a situation from arising. A hash function for which collision cannot be efficiently solved is often said to be collision resistant.

2.1.2 Iterated Hash Functions: We now describe a particular technique by which a compression function, say compress, can be extended to a hash function with an infinite domain, H [18]. A hash function H constructed by this method is called an iterated hash function. We restrict out attention to hash functions whose inputs and outputs are bitstrings (i.e., strings formed of zeros and ones). We denote the length of a bitstring x by $|x|$, and the concatenation of bitstrings x and y is denoted $x \| y$.

Suppose that *compress*: $\{0, 1\}^{m+t} \rightarrow \{0, 1\}^m$ is a compression function (where $t \geq 1$). The evaluation of the iterated hash function H based on compress will consist of three main steps:

Step 1. (Preprocessing) Given an input string x, where $|x| \geq m + t + 1$, construct a string y, using a public algorithm, such that $|x| \equiv 0 \pmod{t}$. Denote $y \equiv y_1 \| y_2 \| \ldots \| y_r$, where $|y_i| = t$ for $1 \leq i \leq r$.

Step 2. (Processing) Let IV be a public initial value which is a bitstring of length m. Then compute the following:

$$\begin{aligned} z_0 &\leftarrow IV \\ z_1 &\leftarrow compress(z_0 \| y_1) \\ z_2 &\leftarrow compress(z_1 \| y_2) \\ &\vdots \quad \vdots \\ z_r &\leftarrow compress(z_{r-1} \| y_r). \end{aligned}$$

Step3. (Optimal output transformation) Let G: $\{0, 1\}^{m+t} \rightarrow \{0, 1\}^l$ be a public function. Define $H(x) = G(z_r)$. The iterated hash function constructed above is

$$H : \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i \rightarrow \{0,1\}^l.$$

A commonly used preprocessing step is to construct the string y in the following way: $y = x \| \text{pad}(x)$, where pad(x) is constructed from x using a padding function. A padding function typically incorporates the value of $|x|$, and pads the result with additional bits (zeros, for example) so that the resulting string y has a length that is a multiple of t.

The preprocessing step must ensure that the mapping $x \mapsto y$ is an injection. (If the mapping $x \mapsto y$ is not one-to-one, then it may be possible to find $x \neq x'$ so that $y \neq y'$.

Then $H(x) = H(x')$, and H would not be collision-resistant.) Note also that $|y| = rt \geq |x|$ because of the required injective property.

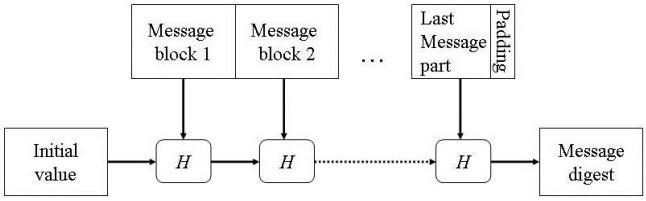

Figure 1. Merkle-Damgaard model for a hash function

The Merkle-Dameggard [16] model for the designs of hash functions has shown its good properties over the years, and in fact many of the existing hash functions follow this model, which is schematically represented in Figure 1.

Essentially, this model simplifies the management of large inputs and the production of a fixed-length output by using a function *H*, which is usually called a compression function. Given a compression function, a hash function can be defined by repeated applications of the compression function until the entire message has been processed. In this process, a message of arbitrary length is broken into blocks whose length depends on the compression function, and padded for security reasons, so the size of the message is a multiple of the block size. The blocks are then hash value for the message which repeatedly operates over a message block and a hash of the previous blocks. The security of this scheme rests on the security of the *H* function, which has been proven to need every property that *H* is supposed to have.

2.1. Cellular Automata

A CA is a collection of simple cells connected in a regular fashion. A CA was originally proposed by John von Neumann as formal models of self-reproducing organisms. Wolfram [11] pioneered the investigation of CA as mathematical models for self-organizing statistical systems and suggested the use of a simple two-state, three-neighborhood (left, self and right) CA with cells arranged linearly in one dimension. The CA structure investigated by Wolfram can be viewed as a discrete lattice of cells where each cell can assume either the value 0 or 1. The next state of a cell is assumed to depend on itself and on its two neighbors (three-neighborhood dependency). The cells evolve in discrete time steps according to some deterministic rule that depends only on local neighbors. In effect, each cell as shown in Figure 2, consists of a storage element (D flip-flop) and a combinational logic implementing the next-state function [19].

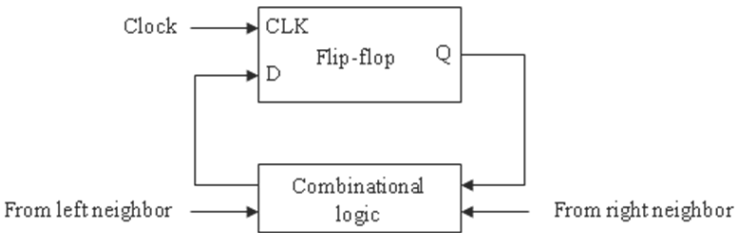

Figure 2. A typical CA cell

In an *m*-state, *k*-neighborhood CA, each cell can exist in *m* different states and the next state of any particular cell depends on the present states of *k* of its neighbors. In this paper, we use a simple 2-state 3-neighborhood CA with the cells in one dimension. The following notations have been used to characterize a CA:

- ♦ *i*: The position of an individual cell in the one-dimensional array of cells
- ♦ *t*: The time step
- ♦ $s_i(t)$: The output state of the *i*th cell at the *t*th time step
- ♦ $S(t)$: The state of the CA at the *t*th instant of time

Mathematically, the next state transition of the *i*th cell can be represented as a function of the present sates of the *i*th, (*i*+1)th and (*i*–1)th cells:

$$s_i(t+1) = f(s_{i-1}(t), s_i(t), s_{i+1}(t))$$

where *f* is known as the rule of the CA denoting the combinational logic.

For a 2-state 3-neighborhood CA, there can be a total of 2^3 distinct neighborhood configurations. For such a CA with cells having only 2 states there can be a total of 256 distinct mappings from all these neighborhood configurations to the next state. If the next-state function of a cell is expressed in the form of a truth table, then the decimal equivalent of the output is conventionally called the rule number for the cell [20].

Table 1. State transition for rule 90, 202, 150 and 232 in 2-state 3-neighbor CA

rule number	111	110	101	100	011	010	001	000
rule 90	0	1	0	1	1	0	1	0
rule 202	1	1	0	0	1	0	1	0
rule 150	1	0	0	1	0	1	1	0
rule 232	1	1	1	0	1	0	0	0

Table 1 specifies four particular sets of transition from a neighborhood configuration to the next state. In Table 1, the top row gives all eight possible states of the three neighboring cells (the left neighbor of the *i*th cell, the *i*th cell itself, and its right neighbor) at the time instant *t*. Rows from second and to fifth give the corresponding states of the *i*th cell at time instant (*t*+1) for four illustrative CA rules. The following logic functions illustrate the rules 90, 202, 150 and 232, where $\oplus$, $\wedge$, and $\vee$ denotes the bitwise XOR, AND, and OR operations.

Rule 90: $s_i(t+1) = s_{i-1}(t) \oplus s_{i+1}(t)$
Rule 202: $s_i(t+1) = (s_{i-1}(t) \wedge (s_i(t) \oplus s_{i+1}(t))) \oplus s_{i+1}(t)$
Rule 150: $s_i(t+1) = s_{i-1}(t) \oplus s_i(t) \oplus s_{i+1}(t)$
Rule 232: $s_i(t+1) = (s_{i-1}(t) \wedge s_i(t)) \vee ((s_{i-1}(t) \vee s_i(t)) \wedge s_{i+1}(t))$

A few definitions used in the characterization of CA are noted below [19].

Definition 1 If the rule of a CA cell involves only XOR logic, then it is called a linear rule. A CA with all the cells having linear rules is called a linear CA, whereas a CA with AND-OR logic is a nonlinear CA.

Definition 2 If in a CA the neighborhood dependence is on XOR, then it is called a non-complemented CA and the corresponding rule is referred as a non-complemented rule. Rules involving XNOR logic are referred to as complemented rules and a CA having a XNOR rules is called a complemented CA.

In 256 rules of 2-state 3-neighorhood CA, 7 rules are only linear, *i.e.*, rule 60, 90, 102, 150, 170, 204, and 240, and there are also 7 rules of their complemented rules, *i.e.*, rule 195, 165, 153, 105, 85, 51, and 15. Thus the remained 242 rules are regarded as nonlinear rules.

Definition 3 A rule is balanced if it contains equal number of 1s and 0s in its 8-bit binary representation; otherwise it is an unbalanced rule.

Definition 4 If in a CA the same rule applies to all the cells, then the CA is called a uniform or regular CA. If in a CA different rules are applied over different cells, then it is called a hybrid CA.

Definition 5 If a state transition of a CA contains only cyclic states, then the CA is called a group CA; otherwise it is a non-group CA. The rule applied on a uniform group CA is called a group rule; otherwise it is a non-group rule.

Definition 6 A CA is said to be a null boundary CA if the left (right) neighbor of the leftmost (rightmost) terminal cell is connected to logic 0-state.

Definition 7 A CA is said to be a periodic boundary CA if the extreme cells are adjacent to each other.

Definition 8 A CA is said to be a intermediate boundary CA if the next state of the left-(right-)most cell depends on itself, its right (left) neighbor, and the one next to (before) it.

Figure 3 shows the CA structure applied rule set <90, 202, 150, 232> based on periodic boundary condition. In this paper, we only consider a periodic boundary condition to apply its cyclic property.

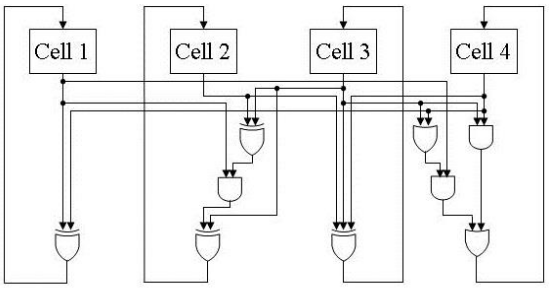

Figure 3. The structure of the periodic boundary CA with the rule set <90, 202, 150, 232>

3. Related Works

This section reviews the previous research for CAs and hash functions.

3.1. Linear and Nonlinear Cellular automata

Numbers of researchers have presented the characteristics of linear CA which can be divided into two parts, group CA and non-group CA [19].

Pris, *et al.*, [21] have reported that complement CA of group CA is also group CA and the proof with the help of a matrix algebraic tools has been reported in [22]. Elspas [23] has characterized the cycle sets in the state-transition diagram of linear machines by considering the factors of their characteristic polynomial. Inverse characterization leading to the works of reversible CA has been reported by Toffoli [24]. A generalized method of a few uniform CA rules and their correlation with length has been reported in [13, 22]. A new type of CA called intermediate boundary CA (IBCA) has been proposed in [25]. For generating high-quality pseudorandom patterns, phase shift analysis of CA has been investigated in [26, 27]

In [28] a special class of non-group CA (referred to as D1*CA) has been utilized for synpaper of easily testable finite-state machines. Non-group CA-based synpaper of easily testable combinational logic has been also addressed in [25]. Another special class of non-group CA used mainly in the field of testable logic synpaper is characterized in [25, 28, 29, 30].

Although the study of linear CA has received considerable attention from researchers due to its linearity and simplicity, the study of nonlinear CA has not received due attention. Recently, however, some interesting properties of such nonlinear CA have been employed successfully in several applications. Ganguly et al. has presented characteristics of nonlinear CA model for pattern recognition [31]. Das, *et al.*, have presented a nonlinear CA based pseudorandom pattern generator for VLSI circuit testing [32-35]. Jeon, *et al.*, have proposed a group and non-group CA based low-complexity authentication scheme and a nonlinear CA based one-time authentication scheme in wireless network [36, 37].

3.2. Cryptographic Hash Functions

A half-dozen years ago, there were several popular cryptographic hash functions from which to choose, including MD5 and SHA-1. MD5 developed by Ron Rivest is a 128-bit hash that is a strengthened form of an earlier Rivest hash function, MD4. Because of the birthday attack, MD5 can only be 64 bits strong; the short length contributed to the National Institute of Standard Technology (NIST) decision not to certify MD5 as a Federal Information Processing Standard (FIPS).

MD5 was already in difficulty in 1993, when Bert den Boer and Antoon Bosselaers found problems with its compression function [38]; further problems were discovered three years later by Hans Dobbertin [39]. The situation became a great deal worse in 2004. At a cryptography meeting in Santa Barbara, Wang, *et al.*, received a standing ovation for work showing collision attacks on MD5 (the attacks also applied to several other hash functions: HAVAL, MD4, and RIPEMD)[40]. There had already been a move away from MD5, but this was the final blow. At the same meeting, Eli Biham and Rafi Chen [41] showed how to find "near" collisions in SHA-0 and Antoine Joux demonstrated an actual collision attack on SHA-0 [42]. SHA-1 still seemed safe.

In 2005 the situation got worse. Wang, in collaboration with Yiqun Lisa Yin and Hongbo Yu, showed a collision attack on SHA-1 that took 2^{69} steps (instead of the

expected 2^{80}) [43]; then Wang, in collaboration with Andrew Yao and Frances Yao, demonstrated a collision attack on SHA-1 that required only 2^{63} steps [44]. The good news is that this was not a second preimage attack and the attack does not mean that all protocols using SHA-1 for integrity were at risk (for example, the usage of SHA-1 in the "handshake" of SSL 3.0/TLS protocol is not affected by these attacks). But it did mean that SHA-1 should be replaced as quickly as possible and should not be used for new applications. How to proceed was both clear and cloudy.

SHA-256, also developed by NSA is waiting in the wings, and is already a FIPS (as are SHA-384 and SHA-512). SHA-256 is three to four times slower than SHA-1, but that is not the real problem. Moving a new algorithm into the infrastructure, whether SHA-256 or a direct SHA-1 replacement (including the established RIPEMD-160, SHA-256 truncated to 160 bits, or a "patched" SHA-1), is not an easy task. Although computer manufacturers understand the importance of replacing SHA-1, and SHA-256 is in the next operating systems being fielded by Microsoft, Sun, and other manufacturers, SHA-1 and MD5 will remain in legacy systems for years to come. And while SHA-256 may share some of the structure of SHA-1 and thus be potentially vulnerable to attack, at 256 bits, the algorithm is large enough, and strong enough, to suffice for now.

3.3. Cryptography with Cellular Automata

Many realistic applications of cellular automata have been reported for pseudo-random [45] and pseudo-exhaustive pattern generation [14], signature analysis [46, 47], fast arithmetic operator design [48-52], error-correcting codes and cryptography [49].
Cellular automata have been used in many cryptographic applications, such as symmetric cryptography, public key cryptography, secret sharing and message authentication.

In symmetric cryptography, the encryption key and the decryption key are the same. The encryption process is based on generation of pseudorandom bit sequences, and a CA can be effectively used for this purpose. CAs for systems with a secrete key were first studied by Wolfram [53], and later by Habutsu, *et al.*, [54], Nandi, *et al.*, [55] and Gutowitz [56]. Tomassini and Sipper [57] proposed one and two dimensional CA for random number generator. Recently, Seredynski, *et al.*, [58] used a CA to design a symmetric key cryptography based on Vernam cipher.

In public-key cryptography, two keys are required: one key is used for encryption and the other for decryption, and one of them is held in private, the other rendered public. Public-key cryptosystems based on CA were proposed by Kari [59].

Secret sharing schemes are cryptographic procedures to share a secret among a set of participants in such a way that only some qualified subsets of these participants can recover the secret. Martín, *et al.*, [60] proposed a secret sharing scheme based on cellular automata and Alvarez, *et al.*, [61] proposed a secure scheme to share secret color images using two dimensional reversible cellular automata.

Dasgupta, *et al.*, [35] proposed a CA based scheme for message authentication. They investigated a particular class of non-group CA that can be employed for generating an efficient message authentication function. Software implementation of the authentication strategy results in 50% reduction of CPU time requirement as compared to MD-5 [4]. An optimal hardware realization of the proposed algorithm will be 10 times faster than any possible parallel realization of MD-5 [20]. Mihaljevic, *et al.*, [31] proposed a family of dedicated one-way hash functions based on linear CA over GF(q)

and Daemen, *et al.*, proposed a framework for the design of one-way functions based on CA [15].

Jeon et al. proposed a Montgomery multiplier architecture and division architecture based on programmable CA in [48, 52]. They also proposed a group and non-group CA based authentication scheme with low-complexity in wireless network in [36]. Recently they applied a non-group CA to role based access control in [62].

4. Cellular Automata based Hash Functions

The first result of the CA application for one-way hash function design has been reported in [16]. The vulnerability of the scheme from [16] is presented together with a result for new CA based hash function called *Cellhash* by Daemen, *et al.*, in 1991 [15]. It assumes preparation of a message so that it is a concatenation of N 32-bit words M_i, $i = 0, 1, \ldots, N-1$, and application of the following procedure:

$H^0 = IV$
$H^j = F(H^{j-1}, M_{j-1}\, M_{j \bmod N} \ldots M_{j+6 \bmod N})$ for $j = 1, 2, \ldots, N-1$
H^N is the hash result,

where $F(H, A)$ is a function with argument H a bitstring of length 257, A is a bitstring of length 256, and IV is the all-zero bitstring of length 257; it returns a bitstring of length 257. $F(H, A)$ consists of five steps as followings:

Step 1. $h_i = h_i \oplus (h_{i+1} \vee (\neg h_{i+2}))$, $\quad 0 \le i < 257$
Step 2. $h_0 = \neg\, h_0$
Step 3. $h_i = h_{i-3} \oplus h_i \oplus h_{i+3}$, $\quad 0 \le i < 257$
Step 4. $h_i = h_i \oplus a_{i-1}$, $\quad 1 \le i < 257$
Step 5. $h_i = h_{10i}$, $\quad 0 \le i < 257$

Step 1 is a nonlinear CA operation where each bitvalue is updated according to the bitvalues in its neighborhood applying a nonlinear updating rule. The nonlinearity of the updating rule has to guarantee the needed confusion. Step 2 consists merely of complementing 1 bit to eliminate circular symmetry in case bitstring A consists of only 0's. Step 3 is a linear CA operation that has to increases the diffusion. Step 4 realizes the actual message bits injection in H to be diffused and confused in subsequent rounds. Step 5 is a bit permutation where bits are placed away from their previous neighbors.

Another major research on a CA based hash function has been reported by Mihaljevic, *et al.*, [17] based on their previous report in [2]. They have proposed a family of fast dedicated one-way hash functions based on linear CA over GF(q) in 1999. Their proposed function follows the model for iterated hash functions, and employs the Davis-Mayer principle. The compression function h is defined by the following:

$$h(M_i, H_{i-1}) = F_{M_i}(H_{i-1}) \oplus H_{i-1},$$

where $F_{M_i}(H_{i-1})$ is a function which maps H_{i-1} according to M_i, and M_i is the ith part of the whole message M. The compression function maps the input variables M_i and H_{i-1} into the output according to the following step 1 thru 5.

Step 1. Nonlinear combining of the M_i and H_{i-1} by the mappings $\{0, 1, \ldots, 2^l-1\} \rightarrow \{0, 1, \ldots, q-1\}$ of H_{i-1} words parameterized by M_i words. Generate an n-dimensional vector X_i with elements $X_{i,k}$, $k = 1, 2, \ldots, n$, from GF(q) according to the following:

$$X_{i,k} = f_{((M_{i,k}+H_{i-1,k}) \bmod 2^l) \bmod K}(M_{i,k}, H_{i-1,k}),$$

where $M_{i,k}$ and $H_{i-1,k}$ are kth word of M_i and H_{i-1} respectively on GF(2^l) for $0 \le k \le n$.

Step 2. (First CA processing) Generate an n-dimensional vector Y_i with elements $Y_{i,k}$ from GF(q):

$Y_i = CA(X_i)$.

Step 3. (Nonlinear mapping and permutation) Generate an n-dimensional vector Y'_i with elements $Y'_{i,k}$ from GF(q) according to the following:

$$Y'_{i,(k+k_0) \bmod n} = \phi_{k \bmod K}((Y_{i,k} + Y_{i,n+1-k}) \bmod q), \quad k = 1, 2, \ldots, n/2$$
$$Y'_{i,(k+k_0) \bmod n} = \phi_{k \bmod K}((Y_{i,k} + Y_{i,k-n/2}) \bmod q), \quad k = n/2 + 1, n/2 + 2, \ldots, n,$$

where k_0 is a certain constant $k_0 < n/2$ and $\phi_j(\,)$ is a nonlinear function which maps an element on GF(q) into an element on GF(q) for $0 \le j \le K-1$, K is the number of functions.

Step 4. (Second CA processing) Generate an n-dimensional vector Z_i with elements $Z_{i,k}$ from GF(q):

$$Z_i = CA(Y'_i).$$

Step 5. (Nonlinear transformation and compression) Generate an n-dimensional vector H'_i with elements $H'_{i,k}$ from GF(2^l) according to the following:

$$H'_{i,k} = \varphi_{k \bmod K}(Z_{i,k}),$$

where and $\varphi_j(\,)$ is a nonlinear function which maps an element on GF(q) into an element on GF(2^l) for $0 \le j \le K-1$. Accordingly, the compression function $h^*(\,)$ is then defined by the following:

$$h^*(M_i, H_{i-1}) = H'_i \oplus H_{i-1} = H_i.$$

Lastly, they use an output function $g^*(H_m)$, where $g^*(\,)$ is defined as a keystream generator.

However the above mentioned scheme in [15] did not provide any specific neighborhood and rules. It means that the scheme is not well defined and designed by CA theory. Meanwhile, CA operations with primitive characteristic polynomial are used only twice in Step 2 and 4 of the compression function in [17], and other nonlinear functions are from HAVAL [7]. Actually it is hard to say that the scheme is a CA based hash function.

Though the mentioned papers have persisted in their security and advantages, they did not provide enough comprehension on security and experimental results. Moreover the previous works did not use specific rules so that it is hard to determine the characteristics of their schemes. In [63], a stronger CA based hash function has been suggested. They used only CA functions to make confusion and diffusion and provided various experimental results.

Step 1. $h_i = h_i \oplus k_i$, $0 \le i \le 255$
Step 2. $h_i = h_{i\text{-}1} \oplus h_i \oplus h_{i+1}$, $0 \le i \le 255$
Step 3. $h_{4i+j} = h_{4i+j\text{-}1} \oplus ((h_{4i+j\text{-}1} \oplus (\neg h_{4i+j})) \vee (h_{4i+j} \oplus h_{4i+j+1}))$, $0 \le i \le 63$, $j = d \bmod 4$
Step 4. $h_i = 3$bits-circular-left-shift(h_i), $0 \le i \le 255$

The scheme employs the MD (Merkle-Dameggard) structure which is well-known as a secure model [16]. The heart of algorithm is a module that consists of processing of 64 rounds. All rounds have the same structure which is composed of XOR operations with the constant *K*, two CA rule functions and 3-bit shift operation. In order to design a concrete hash function, we use combinations of a linear group rule and nonlinear non-group rule. A linear group rule provides a collision resistance from present states to next states and a nonlinear non-group rule provides one-way property and nonlinearity. Rule 150 based on periodic boundary condition is only a linear group rule for a message with 256-bit length, and it has a highest dependency from neighborhood in the middle of the whole linear rules. Meanwhile, we choose rule 23 for a nonlinear non-group CA operation since rule 23 provides not only a high nonlinearity but also a special transition form.

5. Conclusion

In this paper, we had a look around some theories about hash functions and cellular automata. In many studies, they have proposed various hash algorithms for specific environments. In order to use the algorithms on small silicon area, extremely dedicated and tiny architectures are demanded. For satisfying the demand, a CA theory has been used as a basic computational component. In this paper, we summarized typical CA based hash functions. Each scheme has its own pros and cons. But they are still insufficient in minifying and security aspect. Thus well-defined and designed CA based hash function is exceedingly required.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0014977).

References

[1] W. Stallings, “Cryptography and Network Security: Principles and Practice second edition”, Prentice Hall Inc., **(1999).**
[2] M. Mihaljecvic, Y. Zheng and H. Imai, “A Cellular Automaton Based Fast One-Way Hash Function Suitable for Hardware Implementation”, proceeding of PKC’98, LNCS 1431, **(1998)**, pp. 217-233.
[3] R. L. Rivest, “The MD4 message-digest algorithm”, proceeding of Crypto’90, LNCS 537, **(1991)**, pp. 303-311.
[4] R. L. Rivest, “IETF RFC 1321: The MD5 Message-Digest algorithm”, http://www.ietf.org/rfc/rfc1321.txt.
[5] Federal Information Processing Standard (FIPS) Publication 180-2, Secure Hash Standard (SHS), U.S. Doc/NIST, Available from http://csrc.nist.gov/publications/fips/fips180-2/fips 180-2.pdf.
[6] Integrity Primitives for Secure Information Systems: Final Rep. of RACE Integrity Primitives Eval. RIPE-RACE 1040, LNCS 1007, **(1995).**
[7] Y. Zheng, J. Pieprzyk and J. Sebery, “HAVAL - A One-Way Hashing Algorithm with Variable Length of Output”, proceeding of Auscrypt’92, LNCS 718, pp. 83-104 **(1993)**
[8] J. von Neumann, “The Theory of Self-Reproducing Automata”, A. W. Burks, ed., Univ. of Illinois Press, Urbana and London, **(1966)**.
[9] B. Chopart and M. Droz, “Cellular Automata Modeling of Physical Systems”, Cambridge University Press, **(1998).**
[10] T. Toffoli and N. Margolus, “Cellular Automata Machines”, MIT Press, Cambridge, MA, **(1987).**
[11] S. Wolfram, “Statistical Mechanics of Cellular Automata”, Review of Modern Physics, vol. 55, **(1983)**, pp. 601-644.
[12] O. Martin, A. M. Odlyzko and S. Wolfram, “Algebraic Properties of Cellular Automata”, Communications in Mathematical Physics, vol. 93, **(1984)**, pp. 219-258.

[13] A. K. Das and P. P. Chaudhuri, "Efficient Characterization of Cellular Automata", IEE Proceedings, vol. 137, Part E, no. 1, **(1990)**, pp. 81-87.
[14] A. K. Das and P. P. Chaudhuri, "Vector Space Theoretic Analysis of Additive Cellular Automata and Its Applications for Pseudo-exhaustive Test Pattern Generation", IEEE Transactions on Computers, vol. 42, no. 3, **(1993)**, pp. 340-352.
[15] J. Daemen, R. Govaerts and J. Vandewalle, "A Framework for the Design of One-Way Hash Functions Including Cryptanalysis of Damgard's One-Way Function Based on a Cellular Automaton", proceeding of Asiacrypto'91, LNCS 739, **(1993)**, pp. 82-96.
[16] I. B. Damgarrd, "A Design Principle for Hash Functions", proceeding of Crypto'89, LNCS 435, **(1989)**, pp. 416-442.
[17] M. Mihaljecvic, Y. Zheng and H. Imai, "A Family of Fast Dedicated One-Way Hash Functions Based on Linear Cellular Automata over GF(q)", IEICE Transactions on Fundamentals, vol. E82-A, no. 1, **(1999).**
[18] D. R. Stinson, "Cryptography Theory and Practice second edition", Champman & Hall/CRC press, **(2002).**
[19] P. P. Chaudhuri, D. R. Choudhury, S. Nandi and S. Chattopadhyay, Additive Cellular Automata Theory and Applications, vol. 1, IEEE Computer Society Press, **(1997).**
[20] J. Touch, "Performance Analysis of MD5", ACM Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication, **(1995)**, pp. 77-86.
[21] W. Pries, A. Thanailakis and H. C. Card, "Group Properties of Cellular Automata and VLSI Applications", IEEE Transactions on Computers, vol. C-35, **(1986)**, pp. 1013-1024.
[22] A. K. Das, "Additive Cellular Automata: Theory and Application as a Built-In Self-Test Structure", Ph. D. paper, I. I. T., Kharagpur, India, **(1990).**
[23] B. Elspas, "The Theory of Autonomous Linear Sequential Networks", TRE Trans. Circuits, vol. CT-6, **(1959)**, pp. 45-60.
[24] T. Toffoli, "Computation and Construction University of Reversible Cellular Automata", Journal of Computer and System Science, vol. 15, no. 2, **(1977)**, pp. 213-231.
[25] S. Nandi, "Additive Cellular Automata: Theory and Application for Testable Circuit Design and Data Encryption", Ph. D. paper, I. I. T., Kharagpur, India, **(1994).**
[26] P. H. Bardell, "Analysis of Cellular Automata Used as Pseudo-Random Pattern Generators", Int'l Test Conf.'90, **(1990)**, pp. 762-768.
[27] P. D. Hortensius, R. D. McLeod and H. C. Card, "Parallel Pseudo-Random Number Generation for VLSI Systems Using Cellular Automata", IEEE Transactions on Computers, vol. C-38, **(1989)**, pp. 1466-1473.
[28] D. Chowdhury, S. Chakraborty, B. Vamsi and P. Chaudhuri, "Cellular Automata Based Synpaper of Easily and Fully Testable FSMs", proceeding of 6th Int'l Conf. Computer-Aided Design, **(1993)**, pp. 650-653.
[29] S. Chakraborty, D. R. Chowdhury and P. P. Chaudhuri, "Theory and Application of Non-Group Cellular Automata for Synpaper of Easily Testable Finite State Machines", IEEE Transactions on Computers, vol. 45, no. 7, **(1996)**, pp. 769-781.
[30] D. R. Chowdhury, "Theory and Applications of Additive Cellular Automata for Reliable and Testable VLSI Circuit Design", Ph. D. paper, I. I. T., Kharagpur, India, **(1992).**
[31] N. Ganguly, P. Maji, A. Das, B. K. Sikdar and P. P. Chaudhuri, "Characterization of Non-linear Cellular Automata Model for Pattern Recognition", proceeding of AFSS'02, LNAI 2275, **(2002)**, pp. 214-220.
[32] S. Das, D. Dey, S. Sen, B. K. Sikdar and P. P. Chaudhuri, "An Efficient Design of Non-linear CA Based PRPG for VLSI Circuit Testing", proceeding of ASP-DAC'04, **(2004)**, pp. 110-112.
[33] S. Das, A. Kundu and B. K. Sikdar, "Non-linear CA Based Design of Test Set Generator Targeting Pseudo-Random Pattern Resistant Faults", proceeding of the 13th Asian Test Symposium, **(2004)**, pp. 196-201.
[34] S. Das, A. Kundu, S. Sen, B. K. Sikdar and P. P. Chaudhuri, "Non-Linear Cellular Automata Based PRPG Design (Without Prohibited Pattern Set) In Linear Time Complexity", proceeding of the 12th Asian Test Symposium, **(2003)**, pp. 78-83.
[35] P. Dasgupta, S. Chattopadhyay and I. Sengupta, "Theory and Application of Non-group Cellular Automata for Message Authentication", Journal of Systems Architecture, vol. 47, no. 55, **(2001)**, pp. 383-404.
[36] J. C. Jeon, K. W. Kim and K. Y. Yoo, "Low-Complexity Authentication Scheme Based on Cellular Automata in Wireless Network", proceeding of MSN'05, LNCS 3794, **(2005)**, pp. 413-421.
[37] J. C. Jeon, K. W. Kim and K. Y. Yoo, "Non-Group Cellular Automata based One Time Password Authentication Scheme in Wireless Networks", proceeding of MADNES'05, LNCS 4074, **(2006)**, pp. 110-116.
[38] B. D. Boer and A. Bosselaers, "Collisions for the Compression Function of MD5", proceeding of Eurocrypt'03, LNCS 765, **(1994)**, pp. 293-304.
[39] H. Dobbertin, "Cryptanalysis of MD5", Rump Session, Eurocrypt'96, **(1996).**
[40] X. Wang, D. Feng, S. Lai and H. Yu, "Collisions for Hash Functions", Rump Session, Crypto'04, **(2004).**
[41] E. Biham, R. Chen, A. Joux, P. Carrbault, W. Jalby and C. Lemuet, "Collisions in SHA-0 and Reduced SHA-1", proceeding of Eurocrypt'05, LNCS 921, **(2005)**, pp. 36-57.

[42] A. Joux, "Collisions for SHA-0, Rump Session", Eurocrypt'96, **(1996).**
[43] X. Wang, Y. L. Yin and H. Yu, "Finding Collisions in the Full SHA-1", proceeding of Crypto'05, LNCS 3621, **(2005)**, pp. 17-36.
[44] X. Wang, A. Yao and F. Yao, "New Collision Search for SHA-1", Rump Session, Crypto'05, **(2005).**
[45] P. D. Hortensius, R. D. McLeod, W. Pries, D. M. Miller and H. C. Card, "Cellular Automata-based Pseudorandom Number Generators for Built-in Self-test", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 8, no. 8, **(1989)**, pp. 842-859.
[46] A. K. Das, D. Saha, A. R. Chowdhury, S. Misra and P. P. Chaudhuri, "Signature Analyzers based on Additive Cellular Automata", proceeding of 20th International Symposium on Fault Tolerant Computing (FTCS-20), **(1990)**, pp. 265-272.
[47] M. Serra, T. Slater, J. C. Muzio and D. M. Miller, "The Analysis of One-dimensional Linear Cellular Automata and Their Aliasing Properties", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 9, no. 7, **(1990)**, pp. 767-778.
[48] J. C. Jeon, K. W. Kim and K. Y. Yoo, "Evolutionary Hardware Architecture for Division in Elliptic Curve Cryptosystems over GF(2^n)", proceeding of ICNC'05, LNCS 3612, **(2005)**, pp. 348-355.
[49] J. C. Jeon, K. W. Kim, B. H. Kang and K. Y. Yoo, "Cellular Automata Architecture for Elliptic Curve Cryptographic Hardware", proceeding of ICCS'06, LNCS 3993, **(2006)**, pp. 329-336.
[50] J. C. Jeon, K. W. Kim, J. B. Oh and K. Y. Yoo, "Modular Divider for Elliptic Curve Cryptographic Hardware Based on Programmable CA", proceeding of ICCS'06, LNCS 3994, **(2006)**, pp. 661-668.
[51] J. C. Jeon and K. Y. Yoo, "An Evolutionary Approach to the Design of Cellular Automata Architecture for Multiplication in Elliptic Curve Cryptography over Finite Fields", proceeding of PRICAI'04, LNAI 3157, **(2004)**, pp. 241-250.
[52] J. C. Jeon and K. Y. Yoo, "Design of Montgomery Multiplier Architecture Based on Programmable Cellular Automata", Computational Intelligence, vol. 20, no. 3, **(2004)**, pp. 495-502.
[53] S. Wolfram, "Cryptography with Cellular Automata", proceeding of Crypto'85, LNCS 218, **(1986)**, pp. 429-432.
[54] T. Habutsu, Y. Nishio, I. Sasae and S. Mori, "A Secret Key Cryptosystem by Iterating a Chaotic Map", proceeding of Eurocrypt'91, LNCS 547, **(1991)**, pp. 127-140.
[55] S. Nandi, B. K. Kar and P. P. Chaudhuri, "Theory and Applications of Cellular Automata in Cryptography", IEEE Transactions on Computers, vol. 43, no. 12, **(1994)**, pp. 1346-1357.
[56] H. Gutowitz, Cryptography with Dynamical Systems, http://www.santafe.edu/~hag/crypto/crypto.html.
[57] M. Tomassini and M. Sipper, "On the Generation of High-Quality Random Numbers by Two-Dimensional Cellular Automata", IEEE Transactions on Computers, vol. 49, no. 10, **(2000)**, pp.1140-1151.
[58] F. Seredynski, P. Bouvry and A. Y. Zomaya, "Cellular Automata Computations and Secret Key Cryptography", Parallel Computing, vol. 30, no. 5-6, **(2004)**, pp. 753-766.
[59] J. Kari, "Cryptosystems Based on Reversible Cellular Automata", personal communication, **(1992).**
[60] A. M. del Rey, J. P. Mateus and G. R. Sánchez, "A Secret Sharing Scheme based on Cellular Automata", Applied Mathematics and Computation, vol. 170, no. 2, **(2005)**, pp. 1356-1364.
[61] G. Alvarez, A. H. Encinas, L. H. Encinas and A. M. del Rey, "A Secure Scheme to Share Secret Color Images", Computer Physics Communications, vol. 173, no. 1-2, **(2005)**, pp. 9-16.
[62] J. C. Jeon and K. Y. Yoo, "Cellular Automata based Role-Delegation in RBAC", proceeding of ACRI'06, LNCS 4173, **(2006)**, pp. 588-594.
[63] J. C. Jeon, "One-Way Hash Function Based on Cellular Automata", LNEE 215, **(2012)**, pp. 21-28.

Author

Jun-Cheol Jeon

He received B.S. degree in computer engineering from Kumoh National Institute of Technology, Korea in 2000. Being M.S. and Ph. D. degree in computer engineering from Kyungpook National University, Korea in 2003 and 2007 respectively. He was a professor of department of information security at Woosuk University in Korea. He is currently a professor of department of computer engineering at Kumoh National Institute of Technology in Korea. His major research interests include cryptography, crypto-coprocessor design, and quantum cellular automata.